April 2005

Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks

Ashwin Sridharan
University of Pennsylvania

Roch A. Guérin
University of Pennsylvania, guerin@acm.org

Christophe Diot
Intel Research

Copyright 2005 IEEE. Reprinted from *IEEE/ACM Transactions on Networking*, Volume 13, Issue 2, April 2005, pages 234-247.

Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks

Abstract
Traffic engineering is aimed at distributing traffic so as to "optimize" a given performance criterion. The ability to carry out such an optimal distribution depends on the routing protocol and the forwarding mechanisms in use in the network. In IP networks running the OSPF or IS-IS protocols, routing is along shortest paths, and forwarding mechanisms are constrained to distributing traffic "uniformly" over equal cost shortest paths. These constraints often make achieving an optimal distribution of traffic impossible. In this paper, we propose and evaluate an approach that is capable of realizing near optimal traffic distribution without any change to existing routing protocols and forwarding mechanisms. In addition, we explore the trade-off that exists between performance and the overhead associated with the additional configuration steps that our solution requires. The paper's contributions are in formulating and evaluating an approach to traffic engineering for existing IP networks that achieves performance levels comparable to that offered when deploying other forwarding technologies such as MPLS.

Keywords
Networks, Routing, Traffic Engineering, Aggregation

Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks

Ashwin Sridharan, *Student Member, IEEE*, Roch Guérin, *Fellow, IEEE*, and Christophe Diot

***Abstract*—Traffic engineering aims to distribute traffic so as to "optimize" some performance criterion. This optimal distribution of traffic depends on both the routing protocol and the forwarding mechanisms in use in the network. In IP networks running the OSPF or IS-IS protocols, routing is over shortest paths, and forwarding mechanisms distribute traffic "uniformly" over equal cost shortest paths. These constraints often make achieving an optimal distribution of traffic impossible. In this paper, we propose and evaluate an approach that can realize near optimal traffic distribution without changes to routing protocols and forwarding mechanisms. In addition, we explore the tradeoff that exists between performance and the configuration overhead that our solution requires. The paper's contributions are in formulating and evaluating an approach to traffic engineering in IP networks that achieves near-optimal performance while preserving the existing infrastructure.**

***Index Terms*—Aggregation, networks, routing, traffic engineering.**

I. Introduction

As THE AMOUNT and criticality of data being carried on IP networks grows, managing network resources to ensure reliable and acceptable performance becomes increasingly important. Furthermore, this should be accomplished while minimizing or deferring costly upgrades. One of the techniques that is being evaluated by many Internet Service Providers (ISPs) to achieve this goal is traffic engineering. Traffic engineering uses information about the traffic entering and leaving the network to generate a routing scheme that optimizes network performance. Most often the output of traffic engineering is an "optimal" set of paths and link loads that produce the best possible performance given the available resources. However, explicitly setting up such paths and (optimally) assigning traffic to them, typically calls for changes to both the routing protocols and the forwarding mechanism they rely on, e.g., through the introduction of new technology such as MPLS [15].

Currently, two of the most widely used Interior Gateway routing protocols are OSPF [14] and IS-IS [4], and devising solutions that allow these protocols to emulate "optimal routing," is therefore desirable. There are two main difficulties in doing so. The first is that these protocols use shortest path routing with destination based forwarding. The second is that when the protocols generate multiple equal cost paths or next hops for a given destination routing prefix, the forwarding mechanism equally splits the corresponding traffic across them to balance the load. This "equal" splitting is an approximation that depends on the selection of which next hop to use for a given packet. This selection is based either on the value (one for each possible next hop) of a hash function applied to the packet header (source and destination addresses and port numbers), or on the state of a simple round-robin scheme that cycles through the possible next hops. Although the latter option is occasionally used and provides for a more even distribution of load across possible next hops, the former option is the more commonly used in practice. This is because it preserves packet ordering within a flow, and because the large number of flows that modern routers handle results in a good approximation of equal splitting [5]. For simplicity, in the paper we assume that traffic is split in exactly equal fractions across equal cost next hops.

The constraints imposed by the use of both shortest path routing and equal splitting across equal cost paths make it difficult or even impossible to achieve optimal traffic engineering link loads. One of the first works to explore this issue was [7], where a local search heuristic was proposed for optimizing OSPF weights assuming knowledge of the traffic matrix. Ref. [7] showed that in spite of these constraints, properly selecting OSPF weights could yield significant performance improvements. However, the paper also showed that for some topologies, performance could still differ substantially from the optimal solution. Subsequently, a result from linear programming [2, ch. 17, sec. 17.3] was used in [20] to prove that *any* set of routes can be converted into a set of shortest paths based on some link weights that matches or improves upon the performance of the original set of routes. This establishes that the shortest path limitation is in itself not a major hurdle. However, the result of [20] assumes forwarding decisions that are specific to each *ingress–egress* pair, and more importantly, the ability to split traffic in an arbitrary ratio over different shortest paths. Both of these assumptions are at odds with current IP forwarding mechanisms.

Compatibility with destination based forwarding can be achieved through a simple extension to the result of [20], simply by taking advantage of a property of shortest paths [17], or by readjusting the program formulation itself [1].[1] Accommodating the constraint of uneven splitting of traffic

Manuscript received May 1, 2003; revised January 21, 2004; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Roberts. The work of A. Sridharan and R. Guérin was supported in part by the National Science Foundation under Grants ANI-9902943 and ITR-0085930

A. Sridharan and R. Guérin are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: ashwin@ee.upenn.edu; guerin@ee.upenn.edu).

C. Diot is with Intel Research, Cambridge CB3 0FD, U.K. (e-mail: christophe.diot@intel.com).

Digital Object Identifier 10.1109/TNET.2005.845549

[1]This is explained in detail in Section II-A.

across multiple shortest paths is a more challenging task. It is typically not supported with the forwarding path implementations that accompany most routing protocols (some limited support is available with Cisco's proprietary routing protocol EIGRP[2]). A new protocol, OSPF-OMP, was proposed in [19] that circumvents this constraint by modifying the hash function in the router to allow unequal load balancing. However, this will require significant changes to the forwarding mechanisms in the router's data path, which is typically not something that can be accomplished through a simple software upgrade.

The solution we propose leverages the fact that present day routers have thousands of route entries (destination routing prefixes) in their routing table. Instead of changing the forwarding mechanism responsible for distributing traffic across equal cost paths, we plan to control the actual (sub)set of shortest paths (next hops) assigned to routing prefix entries in the forwarding table(s) of a router. *In other words, for each prefix we define a (sub)set of allowable next hops by carefully selecting this subset from the set of next hops corresponding to the shortest paths computed by the routing algorithm.* This allows us to control how traffic is distributed *without* modifying routing protocols such as OSPF or IS-IS, and without requiring changes to the data path of current routers, i.e., their forwarding mechanism. It does, however, require some changes to the control path of routers in order to allow the selective installment of next hops in the forwarding table.

Our initial focus is on gaining a better understanding of how well the selective installment of next hops for different routing prefixes can approximate an optimal traffic allocation (set of link loads). This problem is shown to be NP-complete in [17] and hence one must resort to heuristics. Toward this end, we propose a simple local search heuristic for implementing the solution. We obtain a bound on the gap between the heuristic and optimal allocation, and also demonstrate its efficacy through several experiments. Even though we study the heuristic in the context of a routing problem, we believe that it is generic enough to be of potential use in other load balancing scenarios. The main finding from our investigation is that the performance achieved by this approach is essentially indistinguishable from the optimal.

This being said, an obvious drawback of "hand-crafting" the set of next hops that are to be installed for each routing prefix in a router's forwarding table, is the configuration overhead it introduces. The potential magnitude (proportional to the size of the routing/forwarding table) of this overhead could make this approach impractical. As a result, our next step is to investigate a solution that can help mitigate this overhead, albeit at the cost of a possible degradation in performance. Specifically, we limit the number of routing prefixes for which we perform the proposed selective installment of next hops. Our results indicate that a significant reduction in configuration overhead can be achieved without a major loss of performance.

The rest of the paper is structured as follows. Section II introduces the linear program formulation used in [20] to generate an "optimal" set of shortest paths, and introduces the proposed modifications to make it compatible with existing IP routers. Section III presents a heuristic for approximating an optimal traffic distribution by manipulating the set of next hops assigned to each routing prefix. Section IV presents several experiments that first establish the efficacy of the heuristic of Section III, and then explore the impact on performance of lowering configuration overhead. Section V provides a brief summary of the paper's contributions and outline directions for future work.

[2]Cisco's specification of EIGRP allows unequal load balancing across shortest paths and nonshortest paths.

II. From Optimal Routing to Shortest Path Routing

In this section, we first briefly review the classic result from linear programming [2, ch. 17, sec. 17.3] that was cast in the context of routing in communication networks in [20] to show how optimal routing can be achieved using only shortest paths. We then discuss why this result is not directly usable in current IP networks, and finally propose solutions that allow us to implement the result under the existing paradigm.

The network is modeled as a directed graph $G = (V, E)$ with $v = \mid V \mid$ routers and $e = \mid E \mid$ directed links. We assume the existence of a traffic matrix T where entry $T(s_r, t_r) = d_r$ denotes the average intensity of traffic entering the network at ingress router s_r and exiting at egress router t_r for commodity $r \in \mathcal{R}$. A good reference on how to construct such a traffic matrix can be found in [6]. Assume that an optimal allocation based on some network wide cost function yields a set of paths $\mathcal{P}_r$ for each commodity r (ingress–egress router pair), so that the total bandwidth consumed by these paths on link (i, j) is $\tilde{c}_{ij}$. It can be shown that the same performance, in terms of the bandwidth consumed on each link, can be achieved with a set of shortest paths by formulating and solving a linear program and its dual. The linear program can be formulated as [20]

$$\min \sum_{(i,j)\in E} \sum_{r\in\mathcal{R}} d_r X_{ij}^r$$

subject to

$$\sum_{j:(j,i)\in E} X_{ji}^r - \sum_{j:(i,j)\in E} X_{ij}^r = \begin{cases} 0, & i \neq t_r \\ 1, & i = t_r \end{cases} \quad r \in \mathcal{R}$$

$$\sum_{r\in\mathcal{R}} d_r X_{ij}^r \leq \tilde{c}_{ij}, \quad (i,j) \in E$$

$$0 \leq X_{i,j}^r \leq 1 \quad (i,j) \in E \quad r \in \mathcal{R} \quad (1)$$

where X_{ij}^r is the fraction of traffic for commodity r that flows through link (i, j). Solving the linear program gives a traffic allocation $\{\tilde{X}_{i,j}^r\}$ that consumes no more than $\tilde{c}_{i,j}$ amount of bandwidth on any link (i, j). Note that the formulation is not dependent on the original network cost function used to obtain the paths $\{\mathcal{P}_r\}$. Instead it achieves the same performance by matching the desired bandwidths $\tilde{c}_{ij}$. In order to obtain link weights for shortest path routing, the dual of the linear program as formulated in [20] needs to be solved

$$\max \sum_{r\in\mathcal{R},\ t_r\in V} d_r U_{t_r}^r - \sum_{(i,j)\in E} \tilde{c_{ij}} W_{ij}$$

subject to

$$\begin{aligned} U_j^r - U_i^r &\le W_{ij} + 1 \quad \forall r \in \mathcal{R}, (i,j) \in E \\ W_{ij} &\ge 0 \\ U_{s_r}^r &= 0. \end{aligned} \tag{2}$$

The solution to the dual yields a set of link weights $\{1 + \tilde{W}_{i,j}\}$, from which a set of shortest paths can be constructed.

It is however important to understand that although routing can now be done over shortest paths, this is still quite different from the *forwarding* paradigm currently deployed in existing OSPF and IS-IS networks. The reasons are two-fold and both can be traced to the output of the primal LP, (1), namely, the traffic allocation $\{\hat{X}_{i,j}^r\}$. They govern the fraction of traffic forwarded on each next-hop.

Firstly, observe that the traffic allocation is for each *commodity* or ingress–egress router *pair*. This means that the routing protocol could possibly generate different set of next hops for each ingress–egress pair *on which traffic is to be forwarded*. This would impact the forwarding mechanism on the data path, as the router would need to make decisions on the basis of both ingress and egress routers. Clearly, this is at odds with the current paradigm of destination-based forwarding.

The second problem relates to the fact that current forwarding mechanisms support only equal splitting of traffic on the set of equal cost next hops. The linear program yields a traffic allocation that is not guaranteed to obey this constraint. Modifying the forwarding engine to support unequal splitting (see, for example, [19]) of traffic would involve changes to the data path, namely modification of the hash function to allow for controllable hash boundaries. In the next two subsections we suggest methods that overcome both these problems.

A. Destination Based Aggregation of Traffic

The first problem of translating a traffic allocation that distinguishes between ingress–egress router *pairs* into one that only depends on egress point is relatively straightforward. It can be achieved simply by transforming the individual splitting ratios of ingress–egress pairs that share a common egress router into a possibly different splitting ratio for the aggregate traffic associated with the common egress router. The reason this is possible is because all chosen routes are shortest paths. Shortest paths have the property that segments of shortest paths are also shortest paths, so that once two flows headed to the same egress point meet at a common node they will subsequently follow the same set of shortest paths. This means that we need not distinguish between these packets based on their source address, and can make splitting and forwarding decisions simply based on their destination address.

The aggregation of flows headed to a common egress could either be done *after* the optimal traffic allocation for each commodity has been computed [for example, by the LP in (1)] or in the formulation of the linear program itself. The first method is presented in [17]. The second scheme was presented in [1] and illustrated below. Since traffic allocation is done based only on the egress router, we can aggregate all commodity flow variables headed toward a common egress point and then suitably modify the conservation constraints. A re-formulation of (1) and (2) in the destination aggregated form as presented in [1] is reproduced below.

The primal Linear Program is given by

$$\min \sum_{(i,j)\in E} \sum_{t \in V} Y_{ij}^t$$

subject to

$$\sum_{j:(j,i)\in E} Y_{ji}^t - \sum_{j:(i,j)\in E} Y_{ij}^t = \begin{cases} D^t, & i = t \\ -d_r, & i = s_r, t = t_r \\ 0, & \text{otherwise} \end{cases} \quad \forall \quad i \in V, \quad t \in V$$

$$\sum_{t \in V} Y_{ij}^t \le \tilde{c}_{ij}, \quad \forall (i,j) \in E$$

where

$$D^t = \sum_{r: r \in \mathcal{R}, t_r = t} d_r. \tag{3}$$

The variables of the Primal, $Y_{i,j}^t$, represent traffic on link i, j headed toward egress router t. Also, the flow conservation constraints have been modified to accommodate the aggregation, where D^t represents all traffic headed toward destination t.

The dual can be formulated similar to (2). Let $\mathbf{Z}^t$ represent the vector of node potentials $\{Z_j^t : j \in V\}$ for destination t. Let $\mathbf{P}^t$ represents the right-hand side array in the flow conservation constraints of the primal, (3), that is

$$P^t(i) = \begin{cases} D^t, & i = t \\ -d_r, & i = s_r, t = t_r \\ 0, & \text{otherwise.} \end{cases}$$

The dual is given by

$$\max \sum_{t \in V} \mathbf{P}^t \mathbf{Z}^t - \sum_{(i,j)\in E} \tilde{c_{ij}} W_{ij}$$

subject to

$$\begin{aligned} Z_i^t - Z_j^t &\le W_{ij} + 1 \quad \forall t \in V, (i,j) \in E \\ W_{ij} &\ge 0 \quad \forall (i,j) \in E \\ Z_t^t &= 0 \quad \forall t \in V. \end{aligned} \tag{4}$$

As before, $\{1 + \tilde{W}_{i,j}\}$ still represent link weights for shortest path computation. However, the traffic allocation on each link is for an egress-router and of the form $\{\tilde{Y}_{ij}^t\}$. The above formulation is not only conformal to the paradigm of destination based forwarding, but also reduces the number of variables by a factor of $\mid V \mid$ through removal of information regarding ingress–egress router pairs. This greatly improves the computation complexity involved in obtaining an optimal solution and will be used henceforth throughout the remainder of the paper to compute the optimal traffic allocation and link weights.

B. Approximating Unequal Split of Traffic

In the previous subsection, we saw that solving the problem of source-destination based forwarding decisions was relatively straightforward. Unfortunately, the same does not hold for the

uneven splitting issue, and as mentioned earlier providing such a capability is a significant departure from current operations. Our proposal to overcome this problem is to take advantage of the fact that today's routing tables are relatively large, with multiple routing prefixes associated with the same egress router. By controlling the (sub)set of next hops that each routing prefix is allowed to use, we can control the traffic headed toward a particular egress router(destination). In other words, instead of the current operation that has all routing prefixes use the full set of next hops, we propose to selectively control this choice based on the amount of traffic associated with each routing prefix and the desired link loads for an optimal traffic allocation.

The following example illustrates the idea behind the approach. Assume that at some node, there are four routing prefixes, r_1, r_2, r_3 and r_4 that map to a common destination d and have traffic intensities $t(r_1) = 2$, $t(r_2) = 5$, $t(r_3) = 8$ and $t(r_4) = 4$. Let there be three shortest paths associated with destination d, so that the routing table has 3 possible next hops to d, and assume that the *optimal* distribution of traffic to the three next hops is $f_1 = 6$, $f_2 = 4$ and $f_3 = 9$. We can then intuitively match this traffic distribution by the following next hop assignment: $\{r_1 \rightarrow \text{Hops } 1,\ 3\}$, $\{r_2 \rightarrow \text{Hop } 1\}$, $\{r_3 \rightarrow \text{Hop } 3\}$ and $\{r_4 \rightarrow \text{Hop } 2\}$. The resulting traffic distribution is $f_1' = (2/2 + 5/1) = 6$, $f_2' = (4/1) = 4$, $f_3' = (2/2 + 8/1) = 9$, which matches the optimal allocation.

The advantage of the above approach is that the forwarding mechanism on the data path remains unchanged, as packets are still distributed evenly over the set of next hops assigned to a routing prefix. This means that a close approximation of an optimal traffic engineering solution might be feasible even in the context of existing routing and forwarding technologies. There are, however, a number of challenges that first need to be addressed. The first is the need for traffic information at the granularity of a routing prefix entry instead of a destination (egress router). This in itself is not an insurmountable task as most of the techniques currently used to gather traffic data, e.g., router mechanisms' like Cisco's Netflow or Juniper's cflowd, can be readily adapted to yield information at the granularity of a routing prefix.

The second issue concerns the configuration overhead involved in communicating to each router the subset of next hops to be used for each routing prefix. This can clearly represent a substantial amount of configuration data, as routing tables are large and the information that needs to be conveyed is typically different for each router. The approach we propose and study is to identify a small set of prefixes for which careful allocation of next hops is done and rely on default behavior for the remaining prefixes. The tradeoff will then be in terms of how close one can get to an optimal traffic distribution, while configuring the smallest possible number of routing prefixes. We investigate this tradeoff in Section IV, where we find that near optimum performance can often be achieved by configuring only a small number of routes (routing prefixes).

The third and last challenge is to actually formulate a method for determining which subset of next hops to choose for each routing prefix in order to approximate an optimal allocation. The goal of any solution will be to minimize some metric that measures discrepancy between the optimal traffic allocation and the one achieved under equal-splitting constraints on any hop. In this work, we use the maximum *load* on any hop as a measure of performance, where the *load* on a hop is the ratio of the allocated traffic (by the heuristic) to the optimal traffic. Details regarding the use of another metric, the *gap* between optimal traffic and allocated traffic can be found in [17].

III. Heuristic for Traffic Splitting

Ideally, one should consider the problem of selective next-hop allocation at the global level, that is, do a *concurrent* optimal assignment of next hops for *each* routing prefix at *each* node. However, even the single node allocation problem is NP-complete [17], and hence may not be computationally tractable. Consequently, we propose greedy heuristics that perform independent computations for each routing prefix at each node. These computations are based only on the incoming traffic at the node and the desired outgoing traffic profile. A potential problem with this approach is that the traffic arriving at a node may not match the optimal profile due to the heuristic decisions at some upstream node. Hence, the profile of the outgoing traffic from the node in question, could further deviate from the desired one. However, in our experiments we observe that usually the heuristic is able to track the optimal load profile and hence incoming traffic seen at any node and the resultant outgoing traffic have a near-optimal profile. We have proposed three heuristics that are greedy in nature and try to minimize one of the two metrics mentioned in Section II-B. Since the heuristics are similar in nature, we shall limit our discussion to only one of them, namely the MIN-MAX Load heuristic, because in all our experiments, this heuristic performed the best. Details regarding the other two heuristics can be obtained from [17].

The heuristic we propose works broadly in the following fashion. When performing computation at an arbitrary node:

1) sort routing prefixes destined to a particular egress router in decreasing order of traffic intensity;
2) sequentially assign each routing prefix to a subset of next hops so as to minimize the given metric.

For clarity, we use the following notation in our subsequent discussion.

At an arbitrary router n, when assigning routing prefixes associated with an arbitrary egress router (destination) m to next hops:

1) Denote the set of next hops to egress router m by $\mathcal{K}_{n,m} = \{1, 2, \ldots, K_{n.m}\}$.
2) Denote the desired (optimal) traffic load for egress router m on hop $k \in \mathcal{K}_{n,m}$ by $f_{k,m}$.
3) Denote the collective set of the routing prefixes (at n for m) that need to be assigned to next hops by $\mathcal{X}_{n,m}$. Let $N_{n,m} = \mid \mathcal{X}_{n,m} \mid$. Let traffic intensity of routing prefix $i \in \mathcal{X}_{n,m}$ be equal to x_i.
4) Denote the traffic load on hop k after heuristic H has assigned i routing prefixes by $l_{k,m}^i$. Assume $l_{k,m}^i = 0$ for $i \leq 0$.

A. MIN-MAX Load Heuristic

The MIN-MAX Load heuristic is similar to a work conserving scheduling algorithm which tries to minimize the

maximum load on any processor (the maximum makespan problem, [10]). Heuristic MIN-MAX Load tries to minimize the maximum ratio of assigned traffic to the optimal traffic load over all hops. The difference now is that each task (stream) can be split equally among multiple processors (next hops) and the processors (next hops) can have different speeds (optimal traffic loads).

Algorithm MIN-MAX Load:

1) Sort the set of prefixes $\mathcal{X}_{n,m} :\rightarrow \tilde{\mathcal{X}}_{n,m}$ in descending order of traffic intensity.
2) For each prefix $i \in \tilde{\mathcal{X}}_{n,m}$ choose a subset of next hops $\mathcal{K}^i_{n,m} \in \mathcal{K}_{n,m}$, which minimizes

$$\max_{k \in \mathcal{K}_{n,m}} \left(\frac{l^{i-1}_{k,m} + \dfrac{x_i}{\mid \mathcal{K}^i_{n,m} \mid}}{f_{k,m}} \right).$$

Step 2) can be achieved in two stages. First, for each index $p = 1, 2, \ldots, K_{n,m}$, do a *virtual* assignment of routing prefix i to a set of p hops which yields the smallest maximum. This can be done by simply sorting the set $\{(l^{i-1}_{k,m} + x_i/p)/f_{k,m}\}$, $k \in \mathcal{K}_{n,m}$ in increasing order, re-indexing them and *virtually* assigning i only to the first p hops.

Second, from all the $K_{n,m}$ such possible assignments, choose the one with the smallest maximum for an *actual* assignment. In case of a tie, choose a lexicographically smaller assignment. The complexity of the algorithm is $O(N \log N + NK^2)$, where we have set $N = N_{n,m}$, $K = K_{n,m}$ for simplicity.

We outline our implementation of the heuristics in the pseudo-code below :

```
procedure Selective Hop Allocation
Input ← (Link Weights {1 + W̃_ij}, optimal
traffic allocation {Ỹ^t_ij}, Traffic Matrix T
)
For each destination node m do
 Run Dijkstra's algorithm with weights {1+
W̃_ij}
 For each node n ≠ m in order of de-
creasing distance from m do
  Apply the MIN-MAX Load heuristic to the
set of routing prefixes X_{n,m} to determine,
for each routing prefix i, the set of next
hops K^i_{n,m} ⊂ K_{n,m}
  For each routing prefix i ∈ X_{n,m} do
   Update the intensity of the corre-
sponding routing prefix at each node
j ∈ K^i_{n,m}
  done
 done
done
```

It can be shown that in the single-node case, the MIN-MAX Load heuristic achieves a load ratio that is within a factor of $(1 + \ln K/2)$ of the ratio achieved by an optimum allocation (under the equal splitting constraint). Due to space constraints, the proof has been omitted from text. The interested reader is referred to [17] for details of the proof.

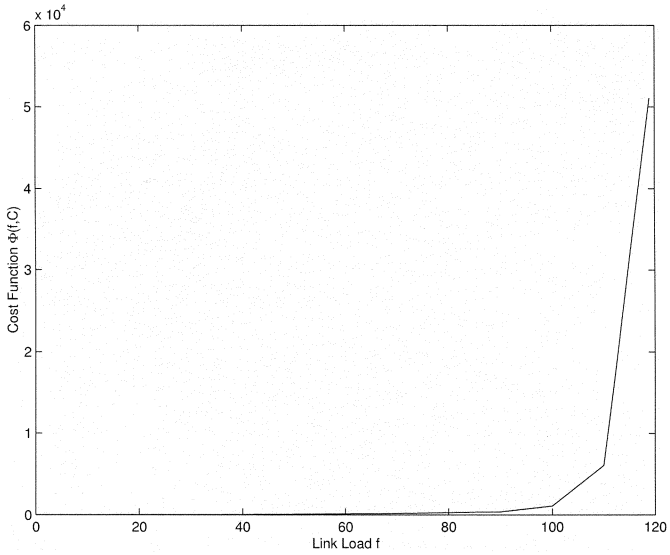

Fig. 1. Evolution of cost function $\Phi_{i,j}(f_{i,j}, C_{i,j})$ as a function of link load $f_{i,j}$.

IV. Experiments

In order to evaluate the effectiveness of our approach, we conducted two sets of experiments on artificially generated topologies as well as on an actual ISP topology, namely the Sprint Backbone. In the first set we studied the performance of our heuristic when compared against optimal routing. In the second set of experiments we studied the tradeoff between performance and configuration overhead by varying the number of routing prefixes for which we controlled the set of next hops they were assigned.

For purposes of comparison, we solved a linear multicommodity flow routing problem with the same piecewise linear cost function used in [7]. The only constraint in the routing problem is flow conservation and consequently it provides a lower bound on the performance of any routing scheme, for the same metric. Hence forth, we shall refer to this problem as the *optimal routing problem* and its solution as the *optimal routing solution*. The solution to this problem is a set of paths (traffic allocation) for each commodity (or destination node) which yields $\tilde{c}_{ij}$, the bandwidth consumed on each link (i, j).

We reproduce the optimal allocation problem with regard to this cost function below for completeness. Let the flow to destination node t on link (i, j) be denoted by $y^t_{i,j}$. Let the total flow on link (i, j) be $f_{i,j} = \sum_{t \in V} y^t_{i,j}$ and the capacity is $C_{i,j}$. Denote the cost of link (i, j) by $\Phi_{i,j}(f_{i,j}, C_{i,j})$, which is a piecewise linear function that approximates an exponentially growing curve [the exact pieces of the function are presented in (6)]. The cost grows as the traffic on the link increases and the rate of growth accelerates with increasing utilization. Evolution

of the cost function with link load is shown in Fig. 1. The *optimal routing problem* may then be formulated as

$$\min \sum_{(i,j)\in E} \Phi_{i,j}(f_{i,j}, C_{i,j})$$

subject to

$$\sum_{j:(j,i)\in E} Y^t_{j,i} - \sum_{j:(i,j)\in E} Y^t_{i,j} = \begin{cases} D_t, & i = t \\ -d_r, & i = s_r \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

$$\forall i \in V, t \in V$$

$$f_{i,j} = \sum_{t\in V} Y^t_{i,j},\ \forall (i,j) \in E$$

$$u_{i,j} = \frac{f_{i,j}}{C_{i,j}},\ \forall (i,j) \in E$$

$$\Phi_{i,j} = f_{i,j},\ u_{i,j} \le \frac{1}{3}$$

$$\Phi_{i,j} = 3f_{i,j} - \frac{2}{3C_{i,j}},\ \frac{1}{3} \le u_{i,j} \le \frac{2}{3}$$

$$\Phi_{i,j} = 10f_{i,j} - \frac{16}{3C_{i,j}},\ \frac{2}{3} \le u_{i,j} \le \frac{9}{10}$$

$$\Phi_{i,j} = 70f_{i,j} - \frac{178}{3C_{i,j}},\ \frac{9}{10} \le u_{i,j} \le 1$$

$$\Phi_{i,j} = 500f_{i,j} - \frac{1468}{3C_{i,j}},\ 1 \le u_{i,j} \le \frac{11}{10}$$

$$\Phi_{i,j} = 5000f_{i,j} - \frac{16\,318}{3C_{i,j}},\ \frac{11}{10} \le u_{i,j} \quad (6)$$

where, as before

$$D^t = \sum_{r:r\in\mathcal{R}, t_r = t} d_r. \quad (7)$$

Note that the approaches presented in [7] and [20] are not limited to any particular cost function. We simply chose this cost function as an example. Also note that the above formulation is based on the idea of destination based aggregation [1] presented in Section II-A and was used for computation because of its lower complexity. In the rest of the section, we explain our experimental set up and discuss our observations regarding performance and complexity tradeoff.

A. Experimental Set Up

For our experiments, the artificial topologies were generated using the Georgia Tech [21] and BRITE [13] topology generators.[3] The topologies constructed using the generators were random graphs chosen from a grid using either a uniform or Waxman distribution. The link capacities were either set uniformly to 500 Mb/s in some cases, or chosen randomly from a uniform distribution in other cases. Actual physical link capacities were used for the topology based on the Sprint backbone.

For the artificially generated topologies, we present results for random traffic matrices that were generated by picking the traffic intensity of each routing prefix from a pareto or uniform distribution. The choice of a pareto distribution was motivated by measurements taken from several routers on the Sprint backbone. We also experimented with other distributions, e.g. uniform, bimodal, exponential and gaussian. Of these, we present results for the uniform distribution since it is representative of the others.

The other parameter of importance is the number of routing prefix associated with each egress router, that is, the granularity of the matrix. For this, we again used both uniform and pareto distributions, as it gives a reasonable coverage for the possible difference in the number of available routing prefixes to a given egress router. The Sprint traffic matrix was based on actual traffic traces downloaded from access links to two of the Sprint backbone routers. The traces was measured at the granularity of the routing table entries[4] and gives us two rows of the traffic matrix. The routing prefix intensities in the remaining rows were generated artificially using a pareto distribution for both intensity and granularity. Details of how the entire Sprint traffic matrix was constructed can be found in [18].

Each experiment was conducted in the following fashion.

1) For each network topology, we generated random traffic matrices, varying both the total number of routing prefixes and distribution[5] from which the ingress traffic intensity of each routing prefix was picked.
2) Hot spots were introduced in the traffic matrix by randomly selecting elements from the traffic matrix and scaling them to create several instances of the traffic matrix. We tested cases where only some of the traffic elements were chosen and also cases where all entries were chosen. In the latter case, this involves scaling the entire traffic matrix.
3) The *optimal routing problem* (7) was then solved for each such instance (topology and traffic matrix).
4) The linear program, (3), with the optimal link bandwidths from the *optimal routing solution* as input, was solved to obtain the traffic allocation (which was aggregated based on destination, ref. Section II-A) and the set of link weights.
5) Finally, our heuristic was run over the network with the link weights and traffic flows from the previous step (*please refer to pseudo-code*) in order to approximate the optimal load profile of (3).

We used ILOG CPLEX to solve the *optimal routing problem* and the linear program. On a Dell 1500 1 GHz machine, it took about 2 hours to solve the *optimal routing problem* and 10 min for the linear program and our heuristic on the largest networks.

B. Performance Comparison Against Optimal Routing

We now present and discuss the results of our experiments. In Fig. 2 we plot cost versus total traffic for the MIN-MAX Load heuristic and the optimal routing solution for the Sprint topology. The horizontal lines represent various levels of maximum average link utilization over all links for optimal routing. The entire traffic matrix was scaled for the experiments involving the Sprint backbone. From the figure, we see that

[3]BRITE allows several options for generating topologies: AS Level, Hierarchical and router level. We chose the router level option.

[4]The routing prefix intensities are averages over 10 hours.

[5]Except in the case of the Sprint traffic matrix.

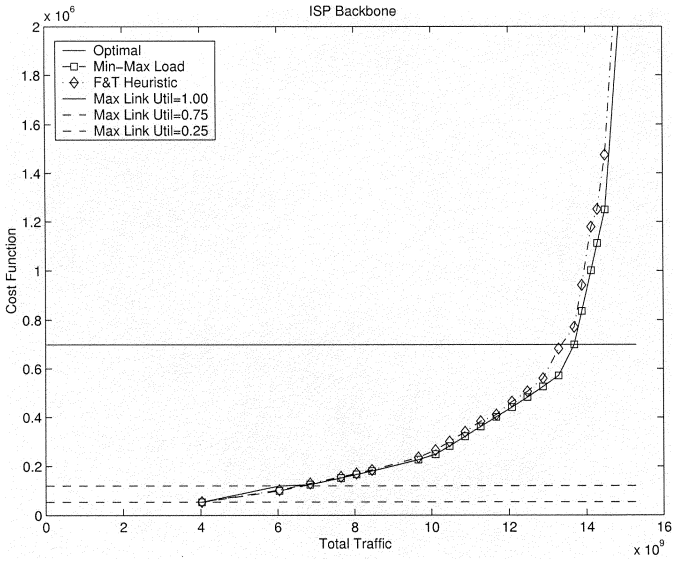

Fig. 2. Sprint backbone: Performance of heuristic versus optimal routing.

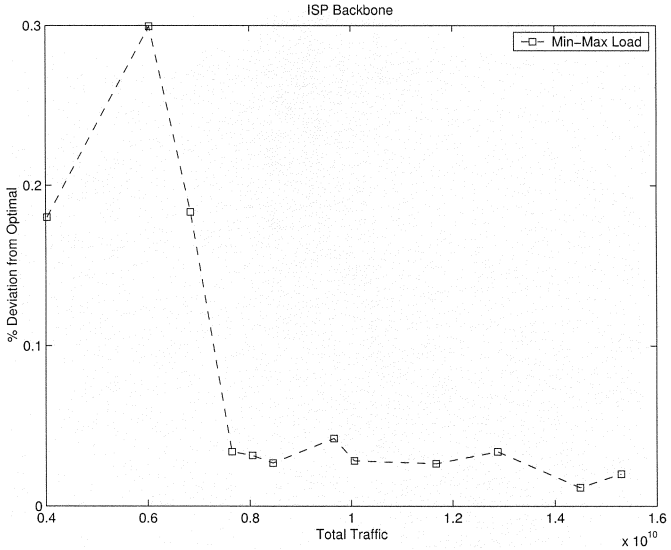

Fig. 3. Sprint backbone: % Deviation of the heuristic from optimal routing.

in all the cases, the heuristic is very near the optimal solution indicating that it is able to match the optimal traffic split very closely.

For comparison, we have also shown the performance of standard OSPF forwarding that requires traffic to be split equally over *all* equal cost next hops. The link weights are computed using our implementation of the heuristic proposed in [7] (denoted by "F&T Heuristic" in the graph). The heuristic uses local search techniques to determine the best OSPF weight setting. Specifically, it searches the neighborhood of a given weight setting by performing random link weight changes in order to determine a better weight setting. It then repeats the search in the neighborhood of the new weight setting. A unique feature of the heuristic is its use of hash tables to avoid cycling as well as re-computation of same routings under different weight settings. In order to avoid getting trapped in local minima valleys, the heuristic performs diversification by randomly selecting a new neighborhood, again by weight perturbation. Similar to [7] we run the heuristic for a fixed number of iterations (5000) and retain the best weight setting.

In our experiments, we found the heuristic to perform quite well, closely tracking the optimal cost for utilizations below 75%. However, instances in [7] show that the heuristic can deviate significantly from the optimal routing even at low utilizations. Furthermore, we note our heuristics can be used to improve the performance of the Fortz and Thorup heuristic (see Section IV-D).

In Fig. 3, we plot the % deviation of the MIN-MAX Load heuristic from the optimal. The maximum deviation of the heuristic is well within 1% of the optimal, highlighting the ability of the heuristic to approximate the desired (optimum) load very closely.

Next, we look at four artificially generated topologies that we used in our experiments. In Fig. 4, we plot Cost versus Total Traffic for the heuristic, optimal routing as well as standard OSPF routing, on a 50 node 200 edge topology with a granularity of 26 500 routing prefixes per node. This number was chosen simply as an approximation of the number of routing

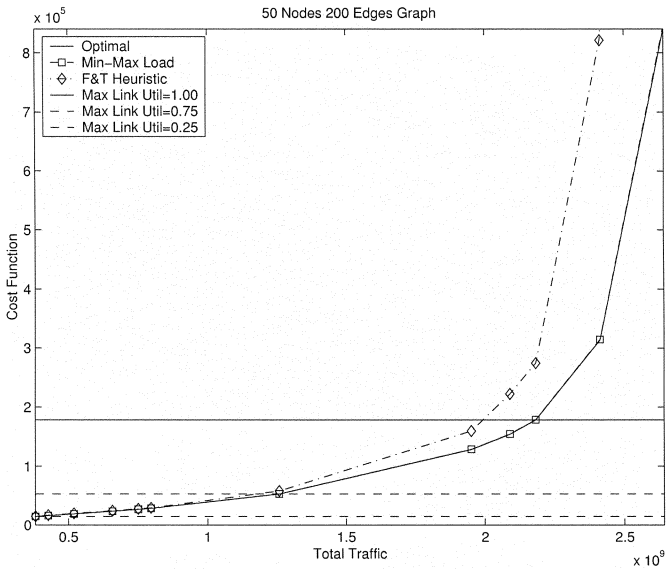

Fig. 4. 50 node 200 edge graph (BRITE generated): Performance of heuristic versus optimal routing.

prefixes in a backbone router. We have conducted experiments with up to 100 000 routing prefixes and as few as 500 routing prefixes without any significant change in performance of the heuristic. The topology was generated using the BRITE generator and all links were set to a capacity of 500 Mb/s. The number of prefixes for each node-pair were chosen from a uniform distribution and the intensity of the entries in the traffic matrix for this experiment were generated from a pareto distribution. Hot-spots were generated by scaling 70% of the traffic elements. We note from Fig. 4, that the heuristic closely tracks the optimal. An alternate view is provided in Fig. 5 where we plot the % deviation of the heuristic from the optimal. The low percentage deviation (0.2%–1%) from the optimal value again highlights the effectiveness of the heuristic.

For the remaining three topologies, the traffic matrix granularity as well as intensities were chosen from a uniform distribution. Hot spots were created by scaling 50% of the node-pairs in the traffic matrix. Cost and % deviation curves for a 60 node

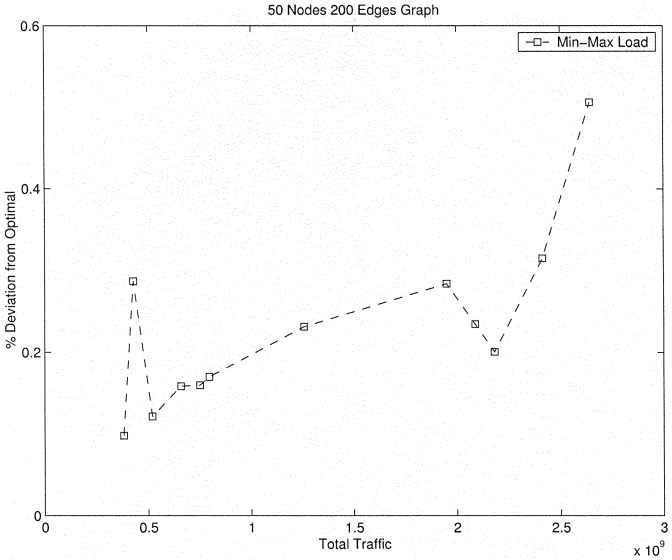

Fig. 5. 50 node 200 edge graph (BRITE generated): % Deviation of the heuristic from optimal routing.

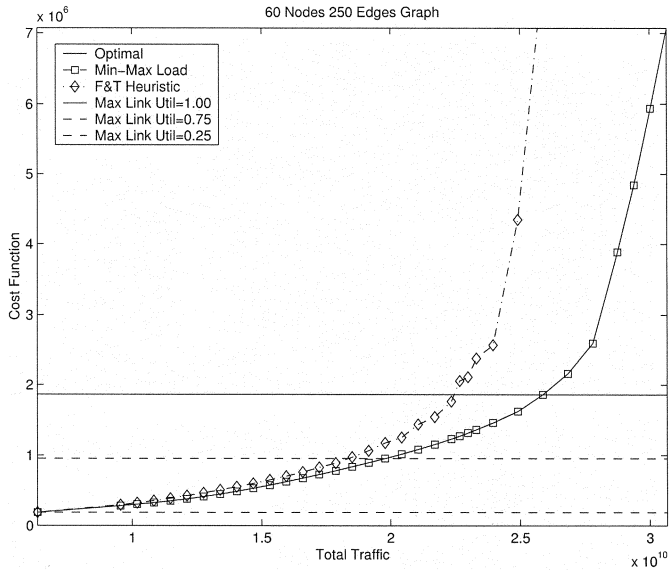

Fig. 6. 60 node 250 edge graph (GT topology generator): Performance of the heuristic versus optimal routing.

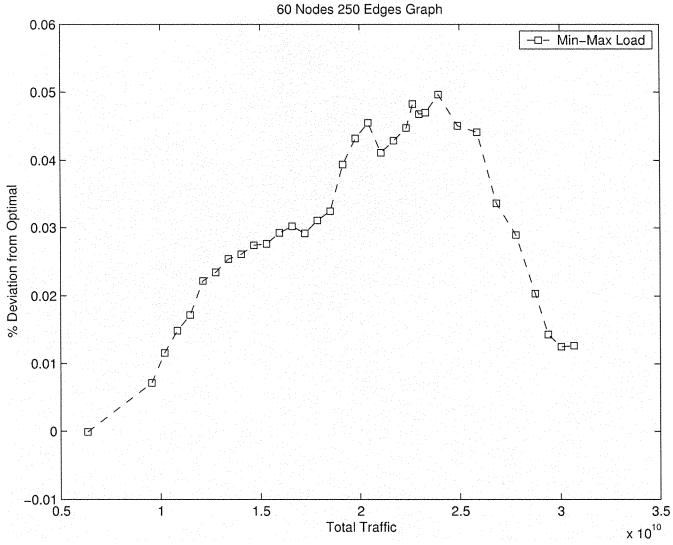

Fig. 7. 60 node 250 edge graph (GT topology generator): % Deviation of the heuristic from optimal routing.

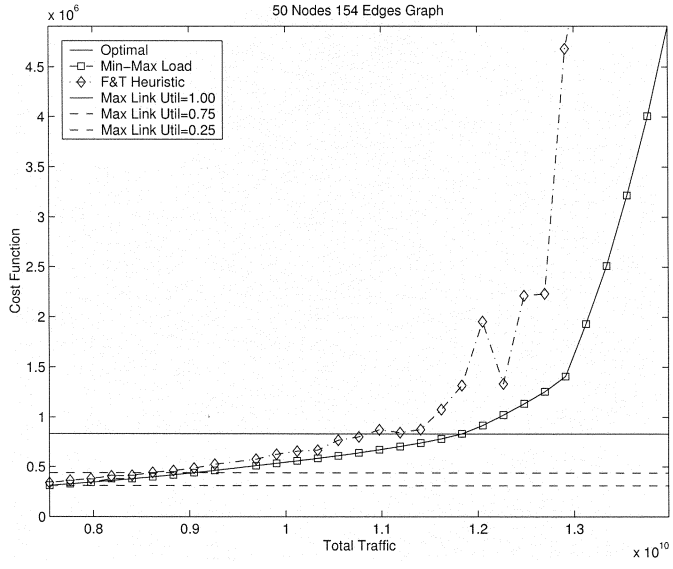

Fig. 8. 50 node 154 edge graph (GT topology generator): Performance of the heuristic versus optimal routing.

250 edge topology are shown in Figs. 6 and 7, respectively. Results for a 50 node 154 edge random topology are presented in Figs. 8 and 9. Both topologies were generated by the Georgia Tech topology generator using the uniform distribution and their link capacities were set to 500 Mb/s. Finally, the cost and % deviation graphs for a 50 node 206 edge random topology generated using the Waxman-1 distribution are shown in Figs. 10 and 11. The link capacities for this topology were randomly chosen from a uniform distribution. We note that for all three topologies, the heuristic performs very well, closely tracking the optimal cost.

C. Noninteger Link Weights

We should point out that the dual of the linear program, (4), is not guaranteed to yield exact integer solutions for link weights. In practice, routing protocols like OSPF and IS-IS have a finite field width for link weight information [11], [16]. If the link weights obtained from the linear program are not exact integers, truncating them to fit within the provided field length can affect performance. This is because the modified link weights can result in routings that are different from the optimal routing. Hence it is important to study how errors in link weights influence performance.

There are two factors that can introduce inaccuracies in link weights. First of course is the loss of precision in rounding off link weights, especially in the presence of recurring fractions, e.g., 2/3. The second factor is the nonzero tolerance required by optimizers to converge to feasible solutions. This implies that the optimal link weights (and path costs) are accurate only within a certain tolerance. For example, two path costs are treated to be equal by the optimizer if the difference in costs is less than the specified tolerance. In our experiments, the tolerance limit was set to 10^{-5}. The presence of recurring fractions

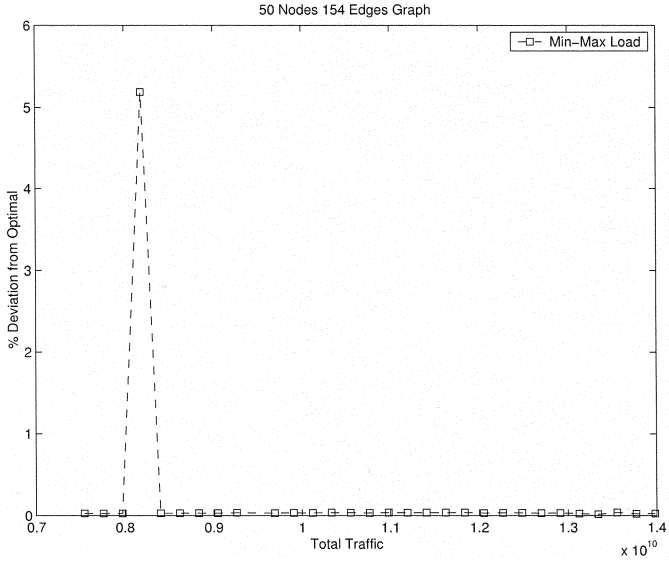

Fig. 9. 50 node 154 edge graph (GT topology generator): % Deviation of the heuristic from optimal routing.

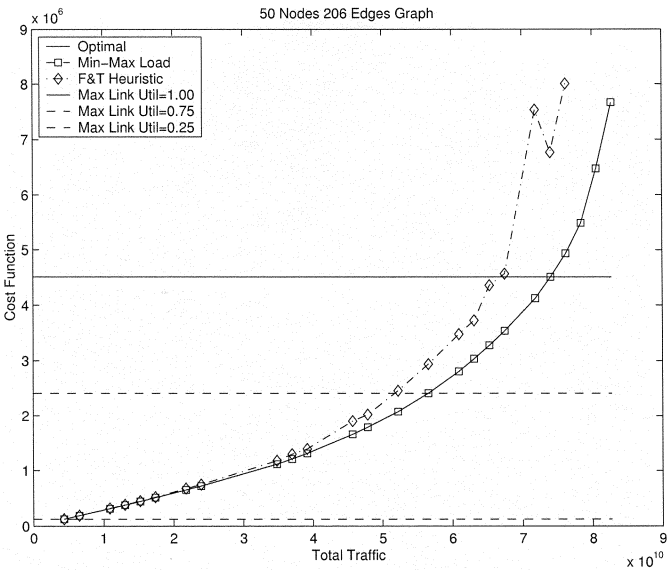

Fig. 10. 50 node 206 edge Waxman graph (GT topology generator): Performance of the heuristic versus optimal routing.

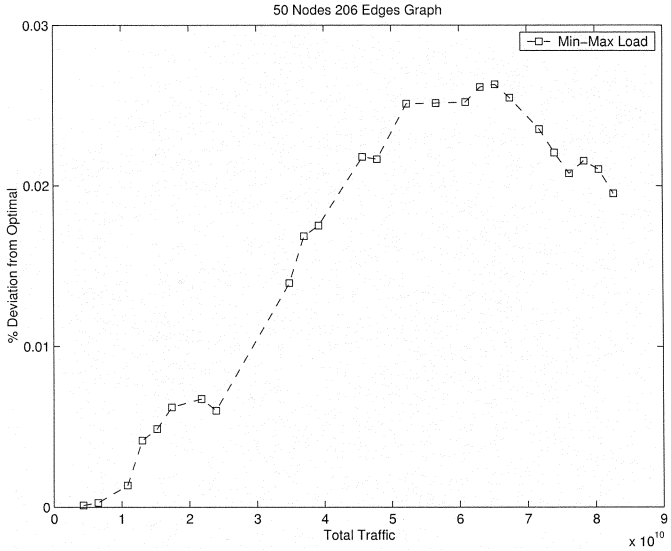

Fig. 11. 50 node 206 edge Waxman graph (GT topology generator): % Deviation of the heuristic from optimal routing.

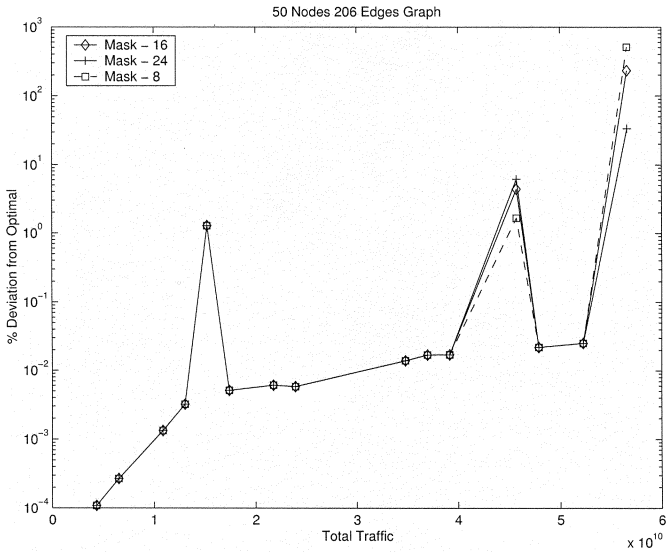

Fig. 12. 50 node 206 edge Waxman topology: Impact of integer weights on performance.

and nonzero tolerance limits means that even with large precision integer representations, these errors still persist and in fact are exacerbated as the integer precision decreases.[6]

Fig. 12 shows the impact of the length of the link weight field on % deviation of the heuristic from optimal cost for the 50 node 206 edge Waxman graph. The link weights are treated as exact integers with precision governed by the field length. We plot curves with field lengths of 8, 16 and 24 bits. Observe that the field-length has little impact on performance till link utilizations start increasing. This is because, in all our experiments, at low utilizations ($< 75\%$), the link weights are almost always exact integers.[7] At higher utilizations, errors due to nonzero tolerances and noninteger link weights come into play so that the performance starts to deviate significantly. Note that even with a 24-bit field length, performance is off by at least 50%, indicating that precision is not the dominant issue. Instead the errors are caused by the inherent inaccuracy in link weights due to nonzero tolerances and noninteger fractions. When treated as exact integers, regardless of the precision used, this produces different routings.

In order to avoid problems due to such errors, we use an approach similar to that used by the optimizer. We treat path costs be to be equal, whenever they differ by less than the specified tolerance thus allowing for inaccuracies. This explains the much better performance (within 3%) seen for the same instance in Fig. 11.

D. Applications of the Next Hop Allocation Technique

Although we have presented our technique for next hop allocation within the framework of shortest path routing as computed by the Linear Program, (3), it is also applicable to paths

[6]Note that since the computer is a finite precision machine, these errors are already present in the link weights obtained from the solution of (4).

[7]At low utilizations, the optimizer can easily find feasible solutions and hence the tolerance limits are not enforced.

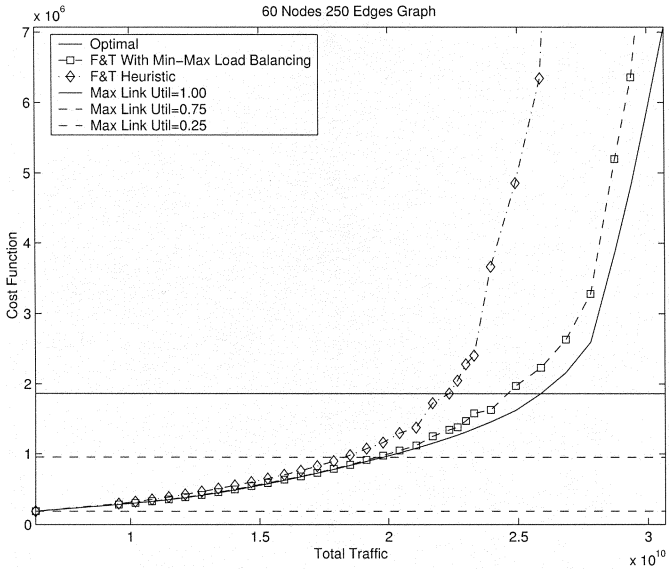

Fig. 13. 60 node 250 edge topology: Impact of MIN-MAX Load heuristic on load balancing with weights computed using Fortz and Thorup heuristic.

computed through other methods. Given a set of paths computed using any other technique, we can find the optimal distribution of traffic over the set of pre-computed paths and then use our next hop allocation technique to match this profile. If the constraining factor is a poor set of paths, then modifying the traffic profile may not offer much improvement. However, if the deciding factor is coarse granularity of load balancing over the paths, significant improvement may be obtained with our heuristic. As an example, one could compute shortest paths using the heuristic proposed by Fortz *et al.* [7]. However, instead of splitting traffic equally over *all* equal cost next hops, it could be distributed in an "optimal fashion" over the computed paths using the next hop allocation technique presented in this paper. In order to achieve this, we first solve the optimal routing problem but with traffic now constrained to flow *only* on paths computed using the link weights obtained from the F&T heuristic. The MIN-MAX Load heuristic is then run to shape the allocated traffic to match the optimal load. We present an example of this procedure for the 60 node 250 edge graph in Fig. 13, were we show the performance improvement of traffic allocation with the MIN-MAX Load heuristic over equal splitting. Note that for both curves, the same set of paths were used, but the MIN-MAX Load heuristic allows for finer load balancing. In other words, the improvement afforded over the standard F&T heuristic by the use of the MIN-MAX Load heuristic can be solely attributed to its ability to better match optimal traffic loads by distributing traffic unevenly. We revisit this issue in the next section, where we explore the impact of the number of equal cost paths (next hops) generated by the routing algorithm.

E. Equal Cost Paths

One of the key features of OSPF/ISIS discussed in this paper is the ability to balance traffic across multiple equal cost paths. In this section, we examine the effectiveness of this feature in improving performance. In other words, how is routing performance dependent on its ability to send traffic on more than one

TABLE I
NUMBER OF EQUAL COST NEXT HOPS FOR EACH INGRESS–EGRESS PAIR

Network	Number of Hops	
	Average	Maximum
Sprint N/W	1.69	5
50 Node, 200 Edge Graph	1.31	4
50 Node, 154 Edge Graph	1.20	4
50 Node, 206 Edge Graph	1.26	4
60 Node, 250 Edge Graph	1.05	5

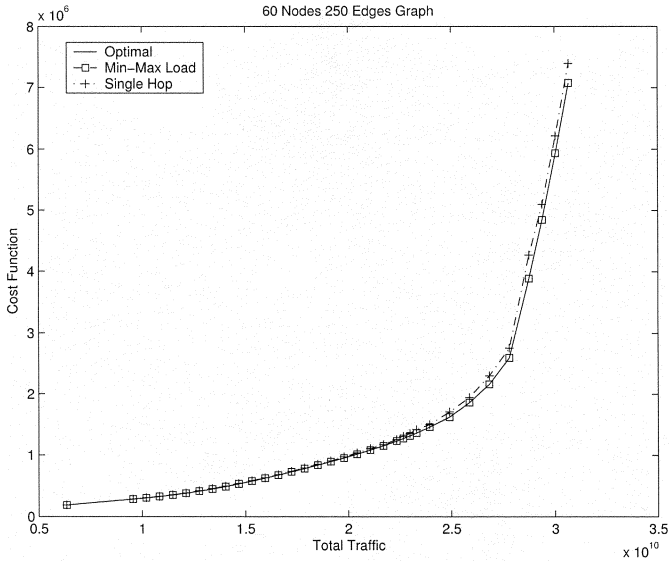

Fig. 14. Impact of equal cost paths on performance for the 60 node 250 edge graph.

path. Intuitively, optimal routing is more likely to use multiple paths to balance traffic at higher loads rather than when capacity is plentiful. Hence, we focus on reasonably high utilization scenarios, albeit less that 100% link loads, for which we compute statistics regarding the number of equal cost next hops used by optimal routing. Table I shows the average and maximum number of equal cost next hops computed by the optimal formulation across all nodes and for all destinations for several network configurations.

We observe that the number of equal cost next hops used by the optimal formulation in order to balance load (although the distribution of traffic across them need not be uniform) varies with the topology. In other words, the benefit of routing over multiple equal cost paths is a function of how the logical topology is designed and also how well the traffic is matched to the topology. To emphasize the dependence of benefits of using equal cost paths on the topology, we evaluate performance when no splitting of traffic is allowed. The "SINGLE HOP" heuristic chooses exactly one next hop, the one with largest "optimal traffic, f_k", for each destination at each node. In all other aspects it functions exactly like the MIN-MAX Load heuristic.

Fig. 14 shows performance curves for the SINGLE HOP heuristic the MIN-MAX Load heuristic on the 60 node 250 edge graph. Observe the near-optimal performance of the SINGLE HOP heuristic, indicating that multiple equal cost paths have little impact on performance. The average number of next hops for the graph in Table I is close to 1, which confirms

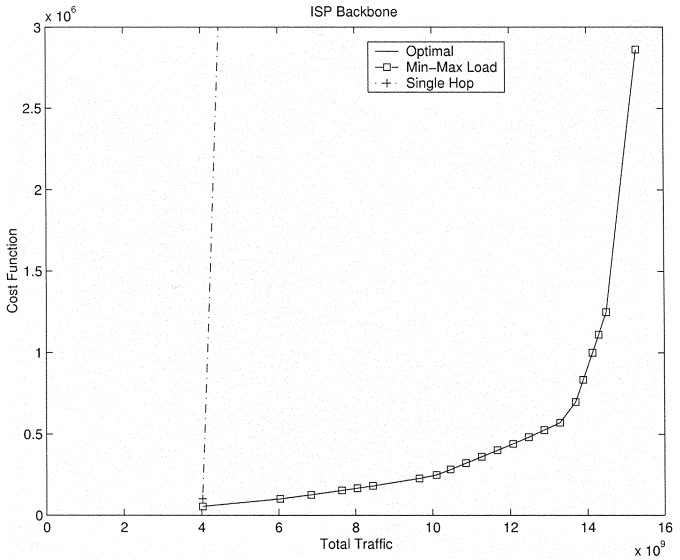

Fig. 15. Impact of equal cost paths on performance for the Sprint backbone.

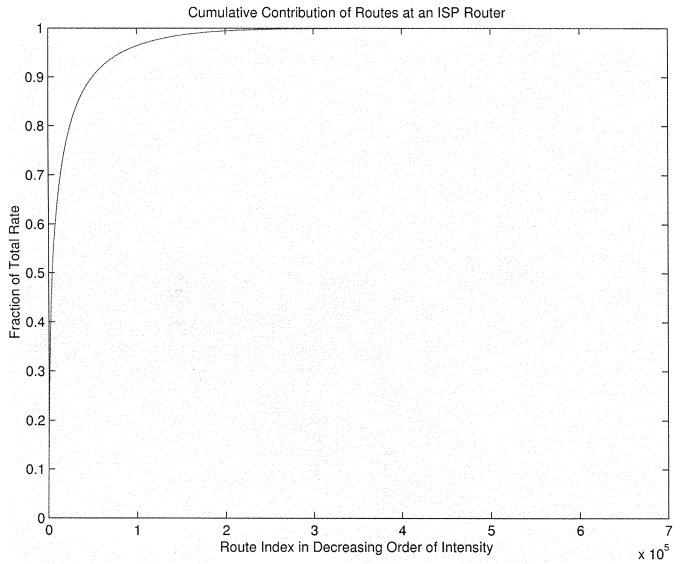

Fig. 16. Cumulative contribution of routing prefixes at a Sprint router sorted in decreasing order of intensity.

our observation. Note that this configuration is the same as that of Fig. 13 that illustrated the improvement that could be achieved by the MIN-MAX Load heuristic over the standard F&T heuristic. This is because whenever the F&T heuristic computes multiple paths, it ends-up allocating traffic equally across them, even when an optimal solution calls for a very uneven traffic allocation. The MIN-MAX Load heuristic, even if it is constrained to using the same paths, is able to generate the uneven loads the optimal routing solution calls for.

In contrast, performance on the Sprint network (Fig. 15) is significantly enhanced by distributing traffic across multiple next hops, and as a result the SINGLE HOP heuristic performs quite poorly. The average number of equal cost next hops for the Sprint network from Table I is close to 2, reflecting this behavior. Indeed, some major ISPs specifically design their logical topologies to increase the number of equal cost paths, especially between geographical hubs, for purposes of both load balancing (as explained in Section I) as well as robustness.

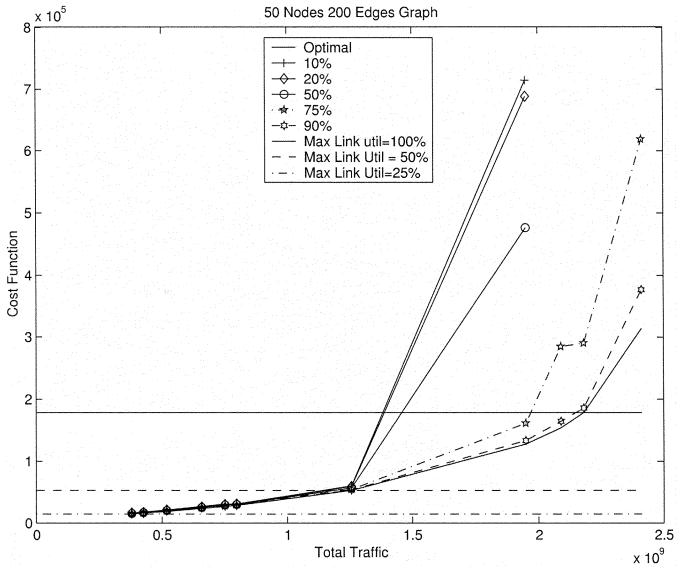

Fig. 17. BRITE 50 node 200 edge topology: Performance as a function of configuration overhead.

TABLE II
CONFIGURATION OVERHEAD: 50 NODE 200 EDGE TOPOLOGY; ALL ENTRIES ARE PER NODE

Prefixes Configured	Total No. of Prefixes	% Allocated Fraction	% of Traffic
75	26500	0.3%	10 %
165	26500	0.62%	20 %
1252	26500	4.7 %	50 %
4500	26500	17.0 %	75 %
11747	26500	44.32 %	90 %

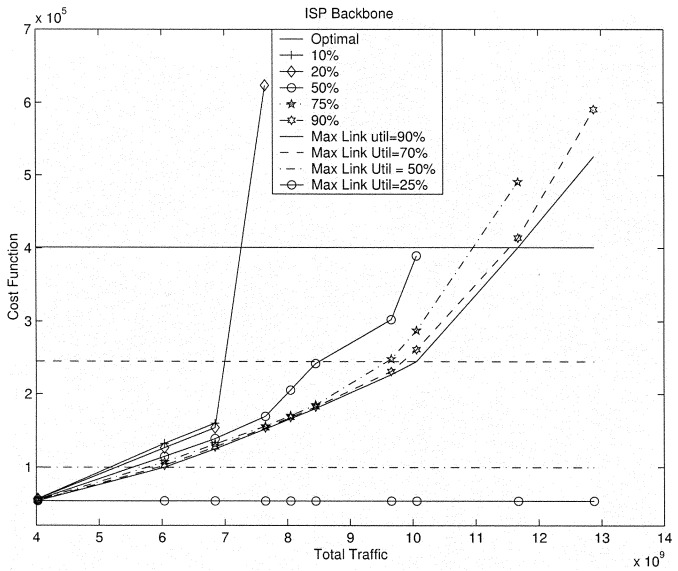

Fig. 18. Sprint backbone: Performance as a function of configuration overhead.

We expect load balancing over equal cost multiple paths to markedly improve performance in these dense networks.

F. Lowering Configuration Overhead

Our other goal was to investigate the tradeoff between configuration overhead and performance. Recall that in the original

TABLE III
CONFIGURATION OVERHEAD: SPRINT BACKBONE, ALL ENTRIES ARE PER NODE

Prefixes configured	Total No. of Prefixes	% Allocated Fraction	% of Traffic
160	30700	0.5%	10 %
200	30700	0.6%	20 %
620	30700	2 %	50 %
1750	30700	6 %	75 %
4180	30700	14 %	90 %

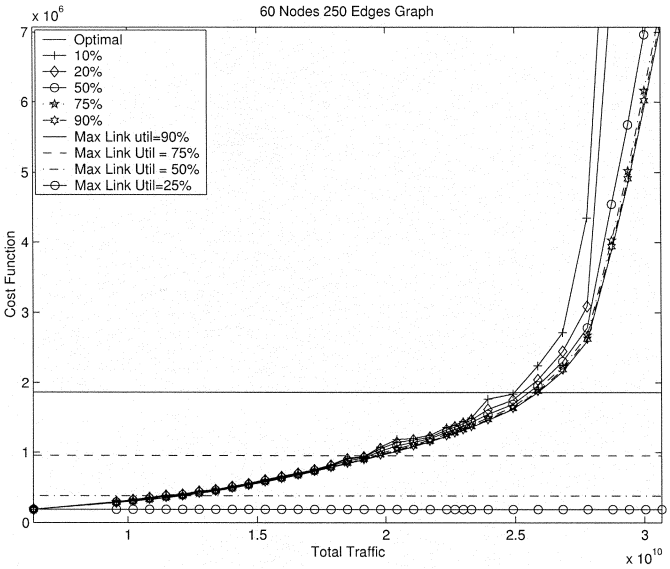

Fig. 19. GT-ITM 60 node 250 edge topology: Performance as a function of configuration overhead.

TABLE IV
CONFIGURATION OVERHEAD: 60 NODE 250 EDGE TOPOLOGY (GT TOPOLOGY GENERATOR), ALL ENTRIES ARE PER NODE

Avg No. of Routes configured	Total No. of Routes	% Allocated Fraction	% of Traffic
5125	117890	4.3%	10 %
14183	117890	12.0%	20 %
32754	117890	27.5 %	50 %
57940	117890	49.1 %	75 %
80500	117890	68.2%	90 %

approach the heuristic decides the *subset* of next hops assigned to *every* routing prefix. However, it has been observed that in practice [3], a large fraction of the traffic is distributed over a relatively small number of routing prefixes. Our analysis of the backbone traces obtained from the Sprint router show that 95% of the total traffic was accounted for by only 10% of the routing prefixes, confirming the results reported in [3]. Fig. 16 highlights this observation, where we have plotted the cumulative traffic intensity as a function of the number of routing prefixes sorted in decreasing order of their traffic intensities. We can potentially exploit such a phenomenon by configuring the set of next hops for only a few selective routing prefixes that carry most of the traffic and allowing the default assignment of all next hops for the remaining routing prefixes. This has the advantage of lowering configuration overhead, but raises the question of how it impacts performance.

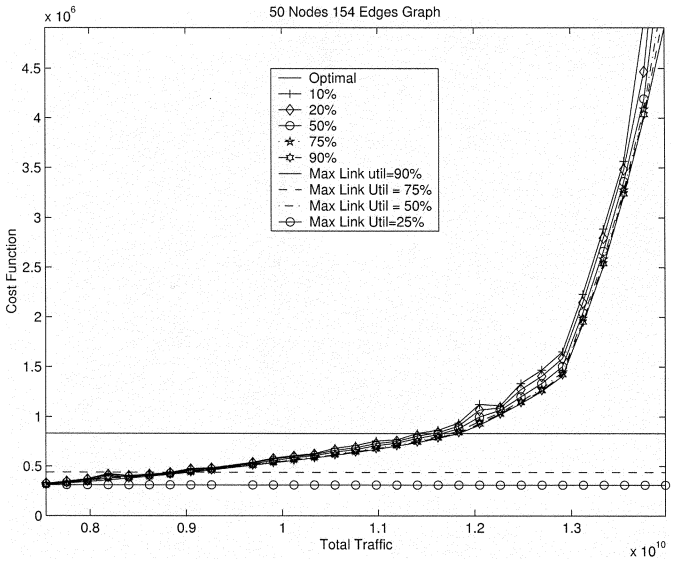

Fig. 20. 50 node 154 edge topology: Performance as a function of configuration overhead.

TABLE V
CONFIGURATION OVERHEAD: 50 NODE 154 EDGE TOPOLOGY; ALL ENTRIES ARE PER NODE

Prefixes configured	Total No. of Prefixes	% Allocated Fraction	% of Traffic
5640	145827	3.8%	10 %
15805	145827	10.8%	20 %
37108	145827	25.4 %	50 %
67132	145827	46 %	75 %
95420	145827	65.4 %	90 %

We carried out a systematic study of such a tradeoff on all the previous topologies. In each instance, we configured the set of next hops at each node for only a certain set of routing prefixes that were selected based on the amount of traffic they carried. The remaining routing prefixes were split equally over the entire set of next hops as would happen with default OSPF/IS-IS behavior. The set of configured routing prefixes was then progressively increased in each experiment to determine the evolution of the impact on performance. In all cases, the MIN-MAX Load heuristic was used when configuring the set of next hops.

The resulting performance curves for the 50 node 200 edge topology are shown in Fig. 17 and the number of configured routing prefixes are shown in Table II. Each curve on the plot is referenced by the amount of traffic that was accounted for by the configured routing prefixes. This can be cross-referenced from the table against the number of routing prefixes that were configured. We observe that on an average, by configuring about 165 routing prefixes per router (which accounts for about 20% of the traffic), we get good performance till about 50% maximum link utilization. If we configure next hops for about 17% of all routing prefixes, or 4500 entries, at a router, we account for approximately 75% of the traffic and the resulting performance is quite close to that of optimal routing.

Experiments conducted on the Sprint Backbone (Fig. 18, Table III) yield similarly encouraging results. We get good performance up to approximately 50%–60% maximum link

TABLE VI
CONFIGURATION OVERHEAD: 50 NODE 206 EDGE WAXMAN TOPOLOGY; ALL ENTRIES ARE PER NODE

Prefixes configured	Total No. of Prefixes	% Allocated Fraction	% of Traffic
4221	97772	4.3%	10 %
11706	97772	11.9%	20 %
27131	97772	27.5 %	50 %
48098	97772	49.8 %	75 %
55754	97772	68.5 %	90 %

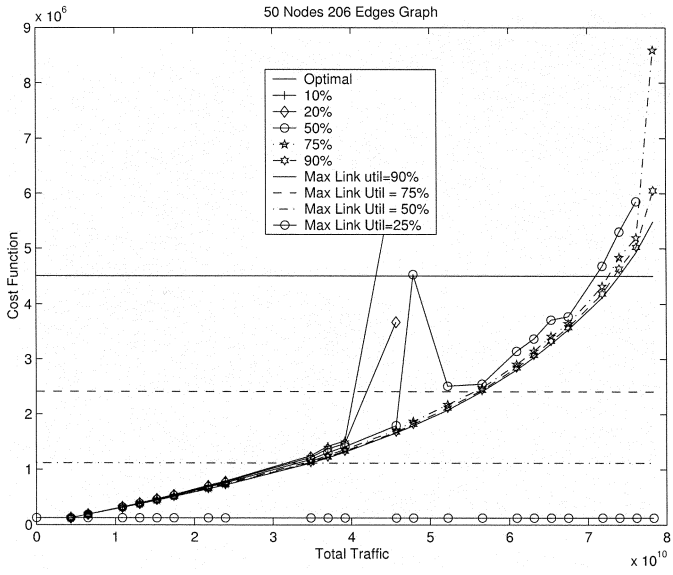

Fig. 21. 50 node 206 edge Waxman topology: Performance as function of configuration overhead.

utilization, by configuring only 200 routing prefixes per router and up to more than 70% link utilization if we configure 600 routing prefixes per router.

The 60 node 250 edge topology (Fig. 19, Table IV) and the 50 node 154 edge topology (Fig. 20, Table V) present an interesting case in that by carefully configuring only 3%–5% of the prefixes we get close to optimal performance over a very wide range of link utilizations. We do not get quite such dramatic results for the 50 node 206 edge Waxman topology (Table VI, Fig. 21), but even in this case, we get good performance by configuring around a quarter of the prefixes (50% of the traffic).

V. CONCLUSION

In this paper, we have described and evaluated an approach that offers the benefits of traffic engineering to IP networks without requiring changes to either the routing protocols or the forwarding mechanisms.

Our contribution is threefold. First, we devised a solution for closely approximating optimal link loads using routing protocols and packet forwarding mechanisms, as they exist today. Second, we proposed a simple heuristic with a provable performance bound (see the Appendix) to implement our solution. We performed several experiments in which the heuristic consistently matched the optimal load profile. We believe the heuristic and those presented in [17] to be general enough to be potentially useful in their own right. Finally, we showed, using actual traffic traces, that configuration overhead can be vastly reduced without significant loss of performance. Specifically, by only configuring next hops for a small set of prefixes, we were able to obtain near-optimal performance for link loads of up to 70%. This is obviously an important aspect for the practical deployment of our traffic engineering solution.

Overall, we believe that the paper provides initial arguments in favor of evolving the current infrastructure to support traffic engineering, if and when needed, rather than embark on a migration to a rather different technology. There may be justifications for such a migration, e.g., better support for policies or VPNs, but traffic engineering does not appear to be one of them, and we hope that the results of this paper can help clarify this issue.

There are several directions in which this work can be extended and further improved. One of them is in dealing with noninteger link weights as discussed in Section IV-C. Although we have been able to successfully deal with this issue by coupling the computational tolerance of the optimizer with the available precision of link weights, we are investigating more general solutions to the problem. Another direction we are currently pursuing is that of making our traffic engineering solution robust to unexpected changes in network topology, e.g., link or router failures. This is obviously an important aspect. One that has been considered by both traffic engineering solutions that rely on new forwarding technologies such as MPLS, e.g., [12], and solutions targeting current IP networks, e.g., [8].

REFERENCES

[1] H. Abrahamsson, B. Ahlgren, J. Alonso, A. Andersson, and P. Kreuger, "A multi path routing algorithm for IP networks based on flow optimization," in *Int. Workshop on Quality of Future Internet Services (QofIS'02)*, Zurich, Switzerland, Oct. 2002.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*. Englewood Cliffs, NJ: Prentice-Hall, 1990, ch. 17, sec. 17.2.

[3] S. Bhattacharya, C. Diot, J. Jetcheva, and N. Taft, "POP-level and access-link level traffic dynamics in a tier-1 POP," in *Proc. ACM SIGCOMM Workshop on Internet Measurement (IMW 2001)*, Nov. 2001, pp. 39–53.

[4] R. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments," Internet Engineering Task Force, Request For Comments (Standard) RFC 1195, 1990.

[5] Z. Cao, Z. Wang, and E. Zegura, "Performance of hashing-based schemes for Internet load balancing," in *Proc. IEEE INFOCOM*, vol. 1, Mar. 2000, pp. 332–241.

[6] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 265–280, Jun. 2001.

[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, vol. 2, Mar. 2000, pp. 519–528.

[8] ——, "OSPF/IS-IS weights in a changing world," *IEEE J. Select. Areas Commun.*, vol. 4, no. 2, pp. 756–767, Feb. 2002.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman, 1979.

[10] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. Appl. Mathemat.*, vol. 17, Mar. 1969.

[11] D. Katz, D. Yeung, and K. Kompella. (2002) Traffic Engineering Extensions to OSPF Version 2. [Online]. Available: draft-katz-yeung-ospf-traffic-09.txt

[12] M. Kodialam and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proc. IEEE INFOCOM*, vol. 2, Mar. 2000, pp. 902–911.

[13] A. Medina, A. Lakhina, I. Matta, and J. Byers. (2001) BRITE: Boston University Representative Internet Topology Generator (Software). Boston Univ., Boston, MA. [Online]. Available: http://cs-www.bu.edu/brite

[14] J. Moy, "OSPF Version 2," Internet Engineering Task Force, Request For Comments (Standard) RFC 2328, 1998.

[15] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 3031, 2001.

[16] H. Smit. (2002) IS-IS Extensions for Traffic Engineering. [Online]. Available: draft-ietf-isis-traffic-04.txt

[17] A. Sridharan, R. Guérin, and C. Diot. (2002) Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks Technical Report. Univ. of Pennsylvania, Philadelphia, PA. [Online]. Available: http://einstein.seas.upenn.edu/publications.html

[18] A. Sridharan, J. Jetcheva, S. Bhattacharyya, C. Diot, R. Guérin, and N. Taft, "On the impact of traffic aggregation on traffic aware routing," in *Proc. 17th Int. Teletraffic Congress*, Brazil, Dec. 2001.

[19] C. Villamizar. (1997) OSPF Optimized Multipath (OSPF-OMP). [Online]. Available: http://www.fictitious.org/omp

[20] Z. Wang, Y. Wang, and L. Zhang, "Internet traffic engineering without full mesh overlaying," in *Proc. IEEE INFOCOM*, vol. 1, Apr. 2001, pp. 565–571.

[21] E. W. Zegura. (1996) GT-ITM: Georgia Tech Internetwork Topology Models (Software). Georgia Tech. [Online]. Available: http://www.cc.gatech.edu/fac/ellen.zegura/gt-itm/gt-itm/tar.gz

Ashwin Sridharan (S'99) received the Bachelors degree in electronic engineering from the Regional Engineering College, Nagpur, India, in 1997 and the M.E. degree from the Indian Institute of Science, Bangalore, in 1999. He is currently pursuing the doctoral degree at the University of Pennsylvania, Philadelphia.

His research interests include traffic engineering and routing algorithms with emphasis on enhancing their performance in existing frameworks.

Roch Guérin (S'85–M'86–SM'91–F'01) received the Engineer degree from ENST, Paris, France, in 1983, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, in 1984 and 1986, respectively.

He joined the Electrical and System Engineering Department, University of Pennsylvania, Philadelphia, in 1998, where he is the Alfred Fitler Moore Professor of Telecommunications Networks. Before joining the University of Pennsylvania, he spent over 12 years at the IBM T. J. Watson Research Center in a variety of technical and management positions. From 2001 to 2004, he was on partial leave from the University of Pennsylvania, starting a company, Ipsum Networks, that pioneered the concept of protocol participation in managing IP networks. He is on the Technical Advisory Board of France Telecom and Samsung Electronics, and has consulted for numerous companies in the networking area. His research has been in the general area of networking, with a recent focus on developing routing and traffic engineering solutions that are both lightweight and robust across a broad range of operating conditions.

Dr. Guérin served as the editor of the ACM SIGCOMM technical newsletter, CCR, from 1998 to 2001. He chaired the IEEE Technical Committee on Computer Communications from 1997 to 1999, and was an editor for the *Journal of Computer Networks*, the *IEEE Communications Surveys*, the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON COMMUNICATIONS, and the *IEEE Communications Magazine*, and a guest editor of a IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS issue on Internet QoS published in December 2000. He served as member-at-large of the Board of Governors of the IEEE Communications Society from 2000 to 2002, as General Chair of the IEEE INFOCOM'98 conference, and as Technical Program co-chair of the ACM SIGCOMM 2001 conference. In 1994 he received an IBM Outstanding Innovation Award for his work on traffic management in the Broad-Band Services Network Architecture. He has been a member of the Association for Computing Machinery (ACM) since 1998.

Christophe Diot received the Ph.D. degree in computer science from INP, Grenoble, France, in 1991.

From 1993 to 1998, he was a research scientist at INRIA, Sophia Antipolis, France, working on new Internet architecture and protocols. From 1998 to 2003, he created and managed the IP research group at Sprint Advanced Technology Laboratory, Burlingame, CA. He recently moved to Intel Research, Cambridge, U.K. He is active in the measurement community, and he has a growing interest in understanding how the Internet will survive mobility and wireless technology.

Dr. Diot has been a member of the Association for Computing Machinery (ACM) since 1996. He has been an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING.